BAB 3

METODOLOGI PENELITIAN

3.1 Analisis Kebutuhan

Penelitian "Implementasi Sistem Pendeteksian Masker Pada Wajah Dengan Menggunakan Arsitektur *EfficientNet*" menggunakan enam buah tahap agar metodologi dan perancangan sistem dapat terpenuhi. Tahap-tahap yang digunakan dalam penelitian ini adalah sebagai berikut :

a. Studi Literatur,

Pada tahap literatur, dilakukan pengumpulan informasi. Informasi yang dikumpulkan berkaitan dengan sistem pendeteksian masker pada wajah dengan menggunakan arsitektur *EfficientNet*, algoritma *EfficientNet*, citra digital, pendeteksian wajah, deep learning dan algoritma *Convolutional Neural Network*. pengumpulan informasi dilakukan dengan membaca dari berbagai sumber seperti jurnal, website, dan juga buku yang terkait dengan informasi yang dibutuhkan

b. Pengumpulan Data,

Data yang dikumpulan dan digunakan adalah data citra wajah manusia sebanyak 2000 gambar citra. Gambar terbagi menjadi 2 label, antara lain yang menggunakan masker dan yang tidak menggunakan masker, serta wajah menghadap kamera dan memiliki perbedaan ekspresi.

c. Analisa dan Perancangan Desain Sistem,

Pada tahap analisa dan perancangan desain sistem, langkah pertama yang dilakukan adalah menganalisa sistem pendeteksian masker pada wajah dengan menggunakan arsitektur *EfficientNet* yang telah ada, mulai dari fitur, cara penggunaan dan bahasa pemrograman yang dilakukan. Kemudian, dilakukan

perancangan terhadap kebutuhan dan terhadap penyesuaian dengan arsitektur *EfficientNet*.

d. Implementasi,

Pada tahap implementasi, dilakukan implementasi dari tancangan yang dilakukan pada tahap analisa dan perancangan desain sistem. Dalam perancangan, digunakan visual code studio versi 1.57.1, python 3.8.5, tensorflow minimal versi 2.5.0 atau lebih tinggi, keras versi 2.4.3, imutils versi 0.5.4, numpy versi 1.19.5, opencv-python versi 4.5.1.*, matplotlib versi 3.4.1, argparse versi 1.4.0, scipy versi 1.6.2, scikit-learn versi 0.24.1, pillo versi 8.2.0, streamlit versi 0.79.0, dan menggunakan bahasa pemrograman Python.

e. Testing dan Perbaikan,

Pengujian hasil sistem pendeteksian masker pada wajah dengan menggunakan arsitektur *EfficientNet* dilakukan secara langsung ketika model sudah dilakukan testing, dan dengan memasukkan citra yang sudah disiapkan. Pengeluaran hasil klasifikasi sesuai dengan label. Proses perbaikan model dapat dilakukan dengan melakukan perbaikan pada tahap pra-proses. Proses pengujian juga dilakukan terhadap aplikasi secara langsung yang diuji dengan mengeluarkan *output* sesuai dengan model klasifikasi.

f. Penulisan Laporan.

Menulis laporan penelitian dari tahap studi literatur hingga pengujian sistem pendeteksian masker pada wajah dengan menggunakan arsitektur *EfficientNet.*

3.2 Perancangan Aplikasi

Penelitian Implementasi Sistem Pendeteksian Masker Pada Wajah Dengan Menggunakan Arsitektur *EfficientNet* menggunakan enam buah tahap agar

metodologi dan perancangan sistem dapat terpenuhi. Tahap-tahap yang digunakan dalam penelitian ini adalah studi literatur, pengumpulan data, analisa dan perancangan desain sistem, implementasi, *testing* dan perbaikan, dan penulisan laporan.

3.2.1 Studi Literatur

Untuk dapat menyelesaikan tahap studi literatur, dilakukanlah proses mencari dan membaca jurnal-jurnal penelitian terkait dengan sistem pengenalan wajah serta CNN. Proses membaca jurnal dilakukan untuk meningkatkan pemahaman terkait pengenalan wajah, citra, deep learning, algoritma CNN, untuk memenuhi landasan teori penelitian.

3.2.2 Pengumpulan Data

Data yang dikumpulkan dan digunakan untuk penelitian adalah data citra wajah manusia yang terdiri dari 2000 citra yang terbagi menjadi 2 model klasifikasi. Model pertama menggunakan masker memiliki sebanyak 1000 citra, dan model kedua yang tidak menggunakan masker memiliki sebanyak 1000 citra.

3.2.3 Analisa dan Perancangan

Proses perancangan aplikasi dilakukan dengan merancang alur kerja dari *EfficientNet*, dan rancangan antarmuka aplikasi berdasarkan kebutuhan.

3.2.4 Desain Sistem

Proses implementasi terbagi menjadi menjadi tiga proses besar, yaitu melakukan pembuatan model klasifikasi dengan menggunakan bahasa pemrograman Python yang dijelaskan pada sub bab metodologi penelitian, pembuatan aplikasi website, dan membuat integrasi Backend antara model,

database, dan Frontend yang telah diciptakan dengan framework Flask Python.

3.2.5 Testing dan Perbaikan

Pengujian hasil pengenalan wajah yang menggunakan masker telah dilakukan dengan menggunakan *pre-trained* model dan telah mengeluarkan hasil klasifikasi sesuai dengan label masing-masing. Proses perbaikan model dilakukan dengan melakukan perbaikan di tahap pra-proses. Proses pengujian juga dilakukan terhadap aplikasi langsung dimana diuji mengeluarkan keluaran sesuai dengan yang model klasifikasikan.

3.2.6 Penulisan Laporan

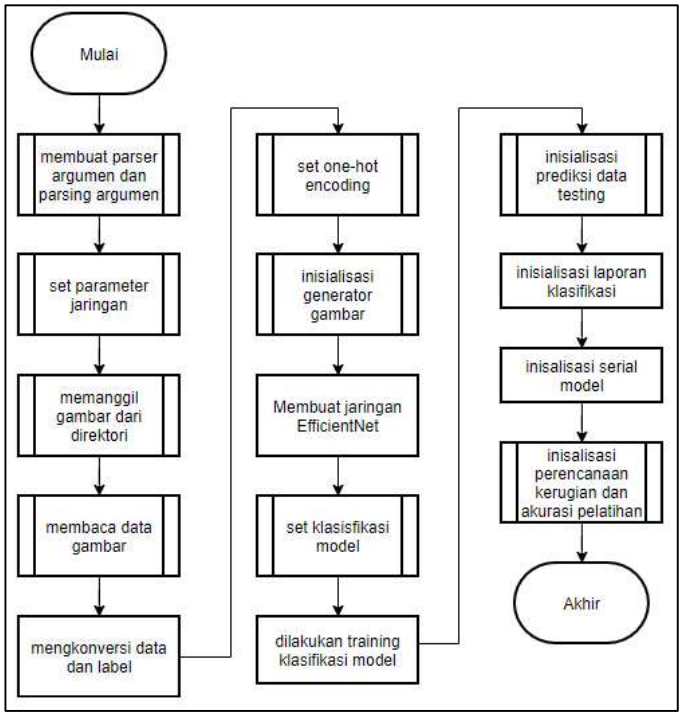
Menulis laporan penelitian dari tahap studi literatur hingga pengujian sistem pengenalan wajah dalam menggunakan masker.

3.3 Implementasi Aplikasi

Berdasarkan hasil dari analisa kebutuhan yang telah dilakukan, dibuat perancangan aplikasi yang dibuat berdasarkan kebutuhan yang dijabarkan. Berikut merupakan rancangan dari aplikasi yang telah dibuat.

3.3.1 Flowchart Utama

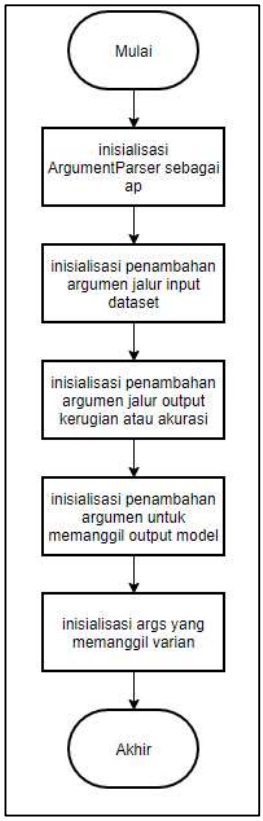
Untuk menyelesaikan penelitian ini, Implementasi sistem pendeteksian masker pada wajah dengan menggunakan arsitektur *EfficientNet* dijelaskan melalui prosedur *flowchart*. Berikut ini adalah *flowchart* yang menggambarkan proses kerja dan implementasi yang dapat dilihat pada Gambar 3.1



Gambar 3.1 Flowchart utama sistem pendeteksian masker pada wajah dengan menggunakan arsitektur *EfficientNet*

3.3.2 Membuat Parser Argumen dan Parsing Argumen

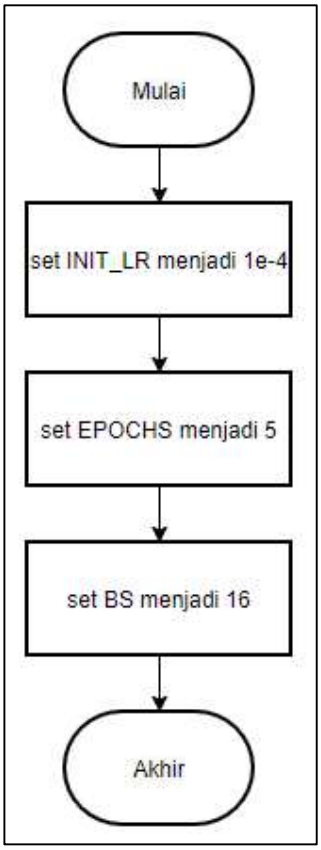
Sesuai dengan Gambar 3.1, pada awal proses adalah dengan mendefinisikan *ArgumenParser* sebagai ap. Kemudian setelah didefinisikan, dilakukan inisialisasi penambahan argumen untuk mencari jalur untuk memasukkan *dataset*. Setelah jalur untuk *dataset* ditemukan, dilanjutkan dengan inisialisasi untuk mendapatkan argumen untuk mendapatkan jalur hasil kerugian atau akurasi dari *plot*. Setelah itu, dilanjutkan lagi dengan penambahan argumen untuk mencari hasil dari model *mask_detector*. Dan yang terakhir adalah dengan menginisialisasikan args dengan memanggil varian dari *ArgumenParser*.



Gambar 3.2 Modul membuat parser argumen dan parsing argumen

3.3.3 Set Parameter Jaringan

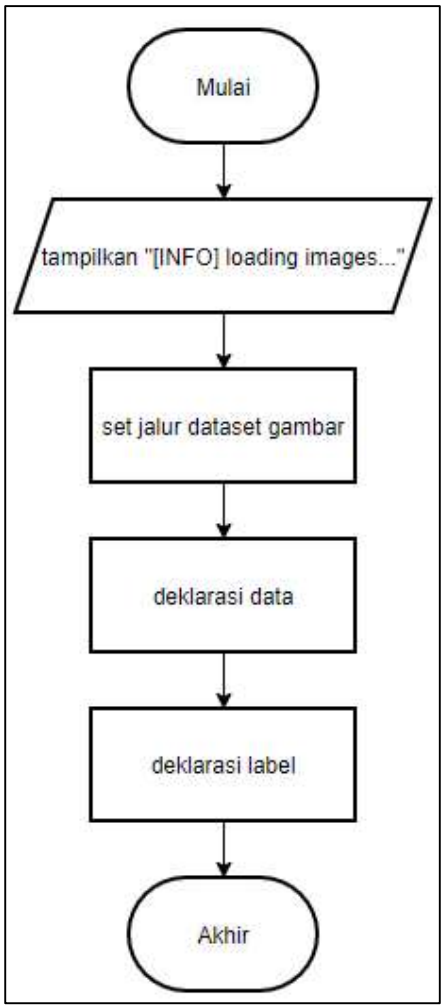
Pada Gambar 3.3 terdapat modul set parameter, yang menginisialisasi kecepatan pembelajaran awal (INIT_LR) menjadi 1e-4. 1e-4 adalah angka yang direpresentasikan dalam notasi ilmiah, bisa juga dianggap dalam bentuk lain yaitu 1 x 10^-4 atau dalam desimal adalah 0,0001. Selanjutnya adalah menginisialisasikan *Epoch* menjadi 5. Dan yang terakhir adalah *batch size* yang disingkan menjadi BS, diinisialisasikan menjadi 16.



Gambar 3.3 Modul set parameter jaringan

3.3.4 Memanggil Gambar dari Direktori

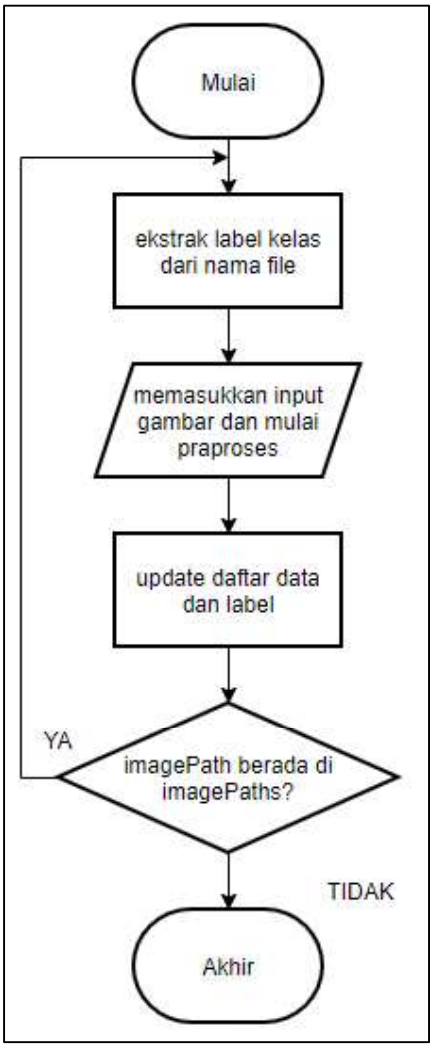
Pada Gambar 4.4 dilakukan inisialisasi untuk print “[INFO] loading images”. Kemudian setelah inisalisasi dilakukan, dilanjutkan dengan set jalur untuk mencari *dataset* gambar. Kemudian deklarasi data dan deklarasi label.



Gambar 3.4 Modul memanggil gambar dari direktori

3.3.5 Inisialisasi Membaca Data Gambar

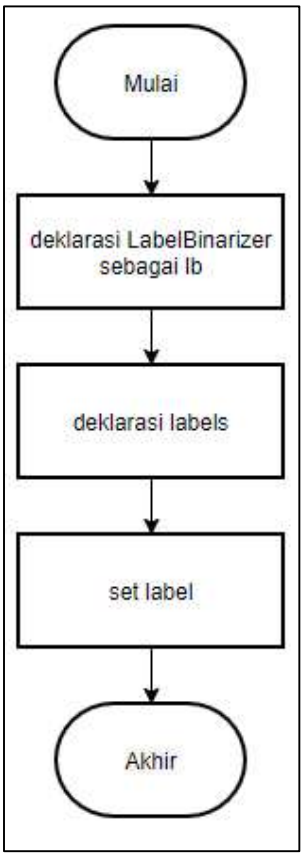
Pada Gambar 3.5, dilakukan pengulangan untuk melakukan ekstrak label dari nama fle, kemudian memasukkan *input*-an gambar dan dimulai praproses gambar tersebut. Dan yang terakhir adalah untuk meng-*update* daftar data dan label yang ada. Pengulangan akan berulang jika *imagePath* masih berada dalam *imagePaths*.



Gambar 3.5 Modul membaca data gambar

3.3.6 Inisialisasi One-Hot Encoding

Pada Gambar 3.6 dilakukan set *one-hot encoding*. Dimana LabelBinarizer direpresentasikan sebagai lb. Kemudian di deklarasikan label agar lb dapat diubah dan cocok dalam label. Dan yang terakhir adalah men-set array label dimulai dari 0.



Gambar 3.6 Modul set *one-hot encoding*

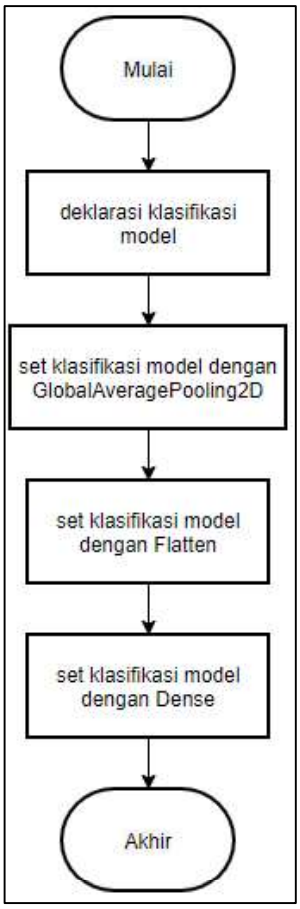
3.3.7 Inisialisasi Generator Gambar

Pada Gambar 3.7 adalah modul untuk membaca gambar, dimana data gambar di-set untuk augmentasi data. Yang dilakukan pertama adalah dengan mendeklarasikan ImageDataGenerator sebagai aug, kemudian men-set yang diperlukan sesuai dengan Gambar 3.8.

Gambar 3.7 Modul membaca data gambar

3.3.8 Inisialisasi Klasifikasi Model

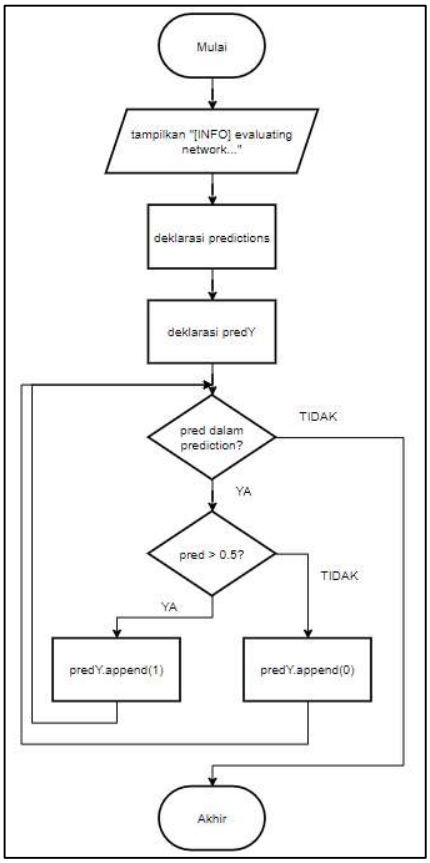
Pada Gambar 3.8 adalah set klasifikasi model, dimana model di klasifikasi, dan men-set klasifikasi model dengan GlobalAveragePool2D, pemerataan (*flatten*), dan kemudian diberikan lapisan (*dense*) yang akan diproses untuk menjadi model.



Gambar 3.8 Modul set klasisfikasi model

3.3.9 Inisialisasi Prediksi Data *Testing*

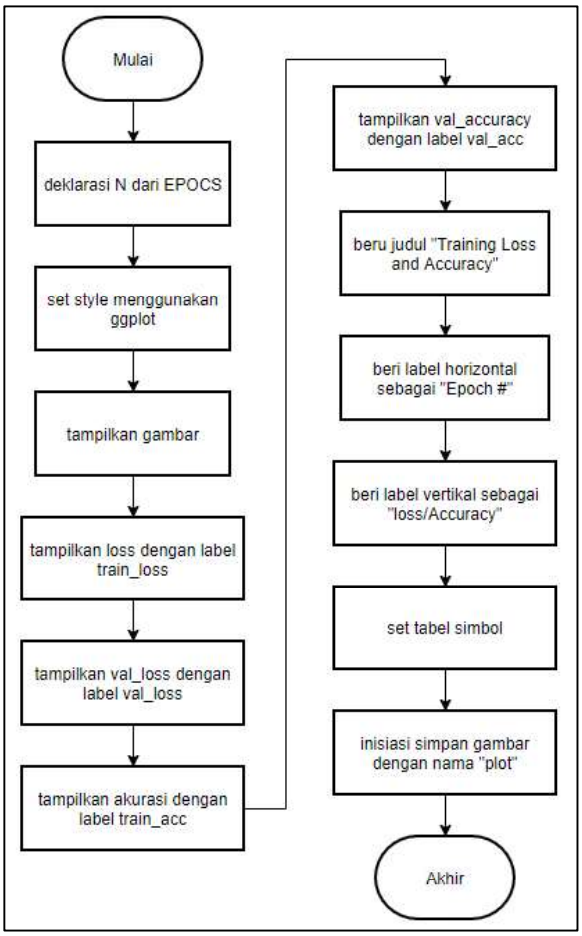
Pada Gambar 3.9 dilakukan inisialisasi prediksi data *testing*. Yang awalnya dilakukan tampilan “[INFO] evaluating network”. Kemudian dilakukan deklarasi *prediction*, dan deklarasi *predY*. Setelah dilakukan deklarasi, ada pengulangan *pred* dalam *prediction*, dimana jika *pred* lebih besar daripada 0.5, maka jika benar maka *predY.append* adalah 1 dan jika tidak maka *predY.append* adalah 0.



Gambar 3.9 Modul inisialisasi prediksi data *testing*

3.3.10 Inisialisasi Perencanaan Kerugian dan Akurasi Pelatihan

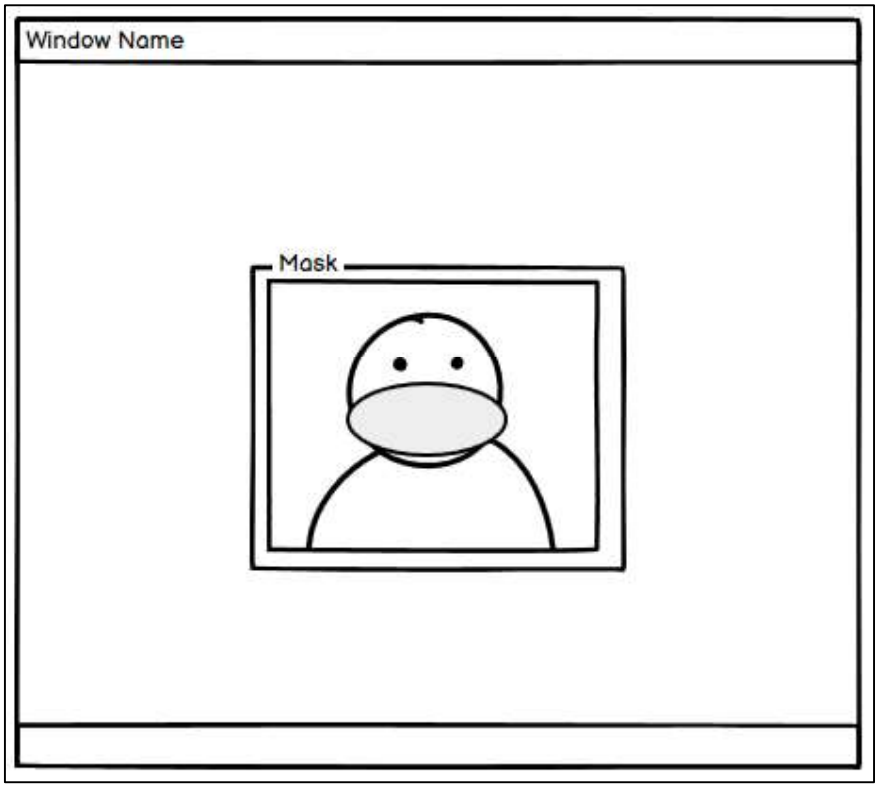
Gambar 3.10 adalah inisialisasi perencanaan kerugian dan akurasi pada saat *training* dilakukan. Yang pertama dilakukan adalah deklarasi N diambil dari EPOCHS. Dan inisialisasi selanjutnya sesuai dengan yang tertera pada Gambar 3.10.



Gambar 3.10 Modul inisalisasi perencanaan kerugian dan akurasi pelatihan

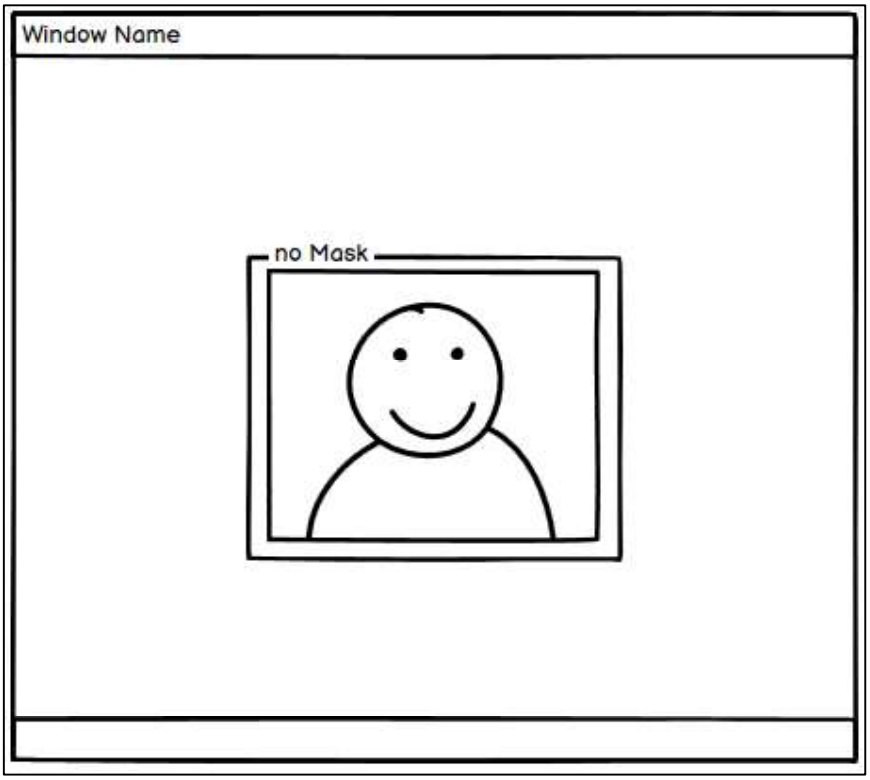
3.4 Rancangan Antarmuka

Desain antarmuka dari sistem pendeteksian masker pada wajah dengan menggunakan arsitektur *EfficientNet* yang digunakan untuk memberikan gambaran tampilan.



Gambar 3.11 Rancang antar muka jika menggunakan masker

Gambar 3.11 merupakan desain antarmuka jika menggunakan masker. Pengguna akan langsung melihat halaman ini, jika pengguna menjalankan sistem. Pada halaman ini, akan langsung mendeteksi apakah pengguna menggunakan masker atau tidak menggunakan masker.



Gambar 3.12 Rancang antar muka jika tidak menggunakan masker

Gambar 3.12 merupakan antarmuka jika menggunakan masker. Pengguna akan langsung melihat halaman ini jika pengguna terdeteksi bahwa pengguna tidak menggunakan masker. Tidak ada *delay* ataupun hal yang perlu pengguna lakukan dalam pergantian Gambar 3.12 menjadi Gambar 3.13, begitu pula sebaliknya. Pengguna hanya perlu melepas masker dan menggunakannya untuk melakukan pengetesan bahwa muka akan terdeteksi menggunakan masker atau tidak.

3.5 Kendala yang Ditemukan

Pada saat *testing*, performa CPU yang digunakan selalu semaksimal mungkin. Sehingga jika menggunakan batch size yang lebih besar, maka kapasitas performa CPU yang tersedia tidak cukup sehingga menghambat performa yang dilakukan.

3.6 Solusi Atas Kendala yang Ditemukan

Diperlukannya spek CPU yang lebih tinggi dalam testing, lebih di rekomendasikan menggunakan PC dibanding menggunakan laptop. Hal ini dikarenakan dalam pemilihan laptop lebih sulit jika ingin memenuhi kriteria, tetapi jika memilih menggunakan PC dapat merancang dan merakit sendiri dalam pemilihan dan penggunaan CPU-nya.